

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

Projet 1 : Création d'un Dashboard pour la visualisation en temps réel des données météorologiques en utilisant Kafka, Spark, Hive, Tableau et Apache Airflow.

Table des matières

I. Objectif du projet	2
II. Prérequis pour le projet - Installation et Configuration	2
1. Environnement de Développement	2
2. Configuration de l'Environnement Dockerisé	2
a. Création des Conteneurs	2
b. Exécution des conteneurs	6
c. Exécution d'un conteneur spécifique	7
3. Installation et Configuration de Tableau	7
a. Téléchargement de Tableau Desktop :	7
b. Activation de Tableau :	8
c. Connexion à Hive depuis Tableau :	8
d. Sélectionner la Table ou la Vue Hive :	10
e. Configuration des Connexions de Données :	11
III. Création du Dashboard pour la visualisation	11
1. Obtention de la clé d'API OpenWeatherMap	11
2. Collecte des Données Météorologiques depuis l'api vers un topic Kafka	12
3. Traitement et Stockage des Données avec Spark et Hive	15
4. Visualisation des Résultats avec Tableau	18
a. Importation des Données	18
b. Création des Visualisations	18
c. Création de Tableaux de Bord (Dashboard)	20
5. Orchestration avec Apache Airflow	21
a. Interface utilisateur	21
b. Création et exécution du dag	22

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

I. Objectif du projet

Ce projet vise à mettre en place un système intégré pour le traitement et la visualisation en temps réel des données météorologiques en utilisant diverses technologies. Il débute par la collecte de données depuis l'API OpenWeatherMap. Ces données transitent ensuite par Kafka pour un traitement en temps réel via Spark Streaming. Les résultats obtenus sont stockés dans Hive, une base de données d'entrepôt dédiée. Enfin, ces données sont exploitées visuellement grâce à Tableau. L'intégralité de ce processus est déployée dans un environnement Dockerisé, simplifiant la gestion et le déploiement de l'infrastructure essentielle à cette chaîne de traitement des données météorologiques.

II. Prérequis pour le projet - Installation et Configuration

Ce projet nécessite l'installation et la configuration préalable de plusieurs outils et environnements. Suivez attentivement ces étapes pour garantir un déroulement sans accroc du projet.

1. Environnement de Développement

- a. Système d'Exploitation : Vérifiez que vous disposez d'un système d'exploitation compatible avec les outils nécessaires (Exemple : Ici, on utilise Windows).
- b. RAM : Assurez-vous d'avoir une mémoire RAM supérieure à 13 Go.
- c. Docker :
 - Téléchargez et installez Docker en suivant les instructions spécifiques à votre système d'exploitation : [lien vers Docker](#)
 - Vérifiez l'installation avec la commande : `docker --version`
- d. Docker Compose :
 - Téléchargez et installez Docker Compose en suivant les instructions spécifiques à votre système d'exploitation : [lien vers Docker Compose](#)
 - Vérifiez l'installation avec la commande : `docker-compose --version`

2. Configuration de l'Environnement Dockerisé

a. Création des Conteneurs

- Élaborez un fichier `docker-compose.yml` décrivant les services nécessaires pour cette application. Se référer au fichier `docker-compose.yml` dans le dossier du projet.

Réalisé par :

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

The screenshot shows a VS Code editor window with a file named `docker-compose.yml` open. The file content is as follows:

```

37 datanode:
38   image: mrugankray/datanode-python:1.0
39   container_name: datanode
40   restart: always
41   volumes:
42     - hadoop_datanode:/hadoop/dfs/data
43     - hadoop_datanode_conda:/root/anaconda
44     - ./configs/datanode_bashrc.txt:/root/.bashrc
45   environment:
46     SERVICE_PRECONDITION: "namenode:9870"
47     CORE_CONF_fs_defaultFS: "hdfs://namenode:9000"
48   ports:
49     - "9864:9864"
50   env_file:
51     - ./hadoop.env
52
53 resource manager:
54   image: mrugankray/resource manager-python:1.0
55   container_name: resource manager
56   restart: always
57   volumes:
58     - hadoop_resource manager_conda:/root/anaconda
59     - ./configs/resource manager_bashrc.txt:/root/.bashrc
60   environment:
61     SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864"
62   ports:
63     - "8088:8088"
64   env_file:
65     - ./hadoop.env
66
67 node manager:
68   image: mrugankray/node manager-python:1.0
69   container_name: node manager
70   restart: always
71   volumes:
72     - hadoop_node manager_conda:/root/anaconda
73     - ./configs/node manager_bashrc.txt:/root/.bashrc

```

The status bar at the bottom indicates the file is in "Restricted Mode" and shows the current cursor position as "Ln 1, Col 1".

```
File Edit Selection View Go Run Terminal Help
Search
docker-compose.yml X
C:\Users\abouelkhir\Documents\ABOU ELKHIR ZITE3\Architecture et Infrastructure Big Data\Big-Data-Cluster-main > docker-compose.yml
1 version: "3"
2
3 services:
4   # Setting Up HDFS & YARN #
5   namenode:
6     image: mrugankray/namenode-spark-airflow-flume-zeppelin:1.1
7     container_name: namenode
8     restart: always
9     ports:
10      - 9870:9870
11      - 9000:9000
12      - 8082:8082 # zeppelin ui
13      - 8080:8080 # spark master web ui
14      - 8081:8081 # spark slave web ui
15      - 4040:4040 # spark driver web ui
16      - 3000:3000 # airflow ui
17     volumes:
18      - hadoop_namenode:/hadoop/dfs/name
19      - hadoop_namenode_conda:/root/anaconda
20      - hadoop_namenode_spark:/opt/spark
21      - hadoop_namenode_zeppelin:/opt/zeppelin
22      - ./configs/zeppelin-site.xml:/opt/zeppelin/conf/zeppelin-site.xml
23      - ./configs/zeppelin-env.sh:/opt/zeppelin/conf/zeppelin-env.sh
24      - ./configs/namenode_bashrc.txt:/root/.bashrc
25      - ./configs/namenode_airflow.cfg:/root/airflow/airflow.cfg
26      - ./dags:/root/airflow/dags
27      - airflow_namenode:/root/airflow
28      - ./configs/namenode/flume/flume-env.sh:/opt/flume/conf/flume-env.sh
29      - ./flume_config/flume.conf:/opt/flume/conf/flume.conf
30      - hadoop_namenode_flume:/opt/flume
31     environment:
32      - CLUSTER_NAME=hadoop-learning
33      - CORE_CONF_fs_defaultTFS-hdfs://namenode:9000
34     env_file:
35      - ./hadoop.env
36
37 datanode:
```

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

```

docker-compose.yml
C:\Users\abouelkhir\Documents\ABOU ELKHIR ZITE3\Architecture et Infrastructure Big Data\Big-Data-Cluster-main> docker-compose.yml
74 environment:
75   SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088"
76   ports:
77     - "8042:8042"
78     - "19888:19888" # to access job history
79   env_file:
80     - ./hadoop.env
81
82 historyserver:
83   image: bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8
84   container_name: historyserver
85   restart: always
86   environment:
87     SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088"
88   ports:
89     - "8188:8188"
90   volumes:
91     - hadoop_historyserver:/hadoop/yarn/timeline
92   env_file:
93     - ./hadoop.env
94
95 # Setting Kafka cluster #
96 zookeeper:
97   image: confluentinc/cp-zookeeper:5.4.0
98   hostname: zookeeper
99   container_name: zookeeper
100   depends_on:
101     - namenode
102     - datanode
103     - resourcemanager
104     - nodemanager
105     - historyserver
106   ports:
107     - "2181:2181"
108   environment:
109     ZOOKEEPER_CLIENT_PORT: 2181
110     ZOOKEEPER_TICK_TIME: 2000
  
```

```

110     ZOOKEEPER_TICK_TIME: 2000
111   volumes:
112     - zookeeper_data:/var/lib/zookeeper/data
113     - zookeeper_log:/var/lib/zookeeper/log
114
115 kafka-broker:
116   image: confluentinc/cp-server:5.4.0
117   hostname: kafka-broker
118   container_name: kafka-broker
119   depends_on:
120     - zookeeper
121   ports:
122     - "29092:29092"
123     - "9092:9092"
124   environment:
125     KAFKA_BROKER_ID: 1
126     KAFKA_ZOOKEEPER_CONNECT: "zookeeper:2181"
127     KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
128     KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka-broker:29092,PLAINTEXT_HOST://localhost:9092
129     KAFKA_METRIC_REPORTERS: io.confluent.metrics.reporter.ConfluentMetricsReporter
130     KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
131     KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
132     KAFKA_CONFLUENT_LICENSE_TOPIC_REPLICATION_FACTOR: 1
133     CONFLUENT_METRICS_REPORTER_BOOTSTRAP_SERVERS: kafka-broker:29092
134     CONFLUENT_METRICS_REPORTER_ZOOKEEPER_CONNECT: zookeeper:2181
135     CONFLUENT_METRICS_REPORTER_TOPIC_REPLICAS: 1
136     CONFLUENT_METRICS_ENABLE: "true"
137     CONFLUENT_SUPPORT_CUSTOMER_ID: "anonymous"
138   volumes:
139     - kafka_broker:/var/lib/kafka/data
140
141 schema-registry:
142   image: confluentinc/cp-schema-registry:5.4.0
143   hostname: schema-registry
144   container_name: schema-registry
145   depends_on:
146     - zookeeper
  
```

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

```

146 - zookeeper
147 - kafka-broker
148 deploy:
149   restart_policy:
150     condition: on-failure
151     delay: 5s
152     max_attempts: 3
153     window: 120s
154 ports:
155   - "8083:8083"
156 environment:
157   # SCHEMA_REGISTRY_KAFKASTORE_BOOTSTRAP_SERVERS: http://kafka-broker:29092
158   SCHEMA_REGISTRY_HOST_NAME: schema-registry
159   # setting SCHEMA_REGISTRY_KAFKASTORE_CONNECTION_URL to zookeeper. schema registry will get kafka server url from zookeeper
160   SCHEMA_REGISTRY_KAFKASTORE_CONNECTION_URL: "zookeeper:2181"
161   SCHEMA_REGISTRY_LISTENERS: http://schema-registry:8083
162   SCHEMA_REGISTRY_KAFKASTORE_ZK_SESSION_TIMEOUT_MS: 300000
163   SCHEMA_REGISTRY_KAFKASTORE_INIT_TIMEOUT_MS: 300000
164 # Setting up Hive #
165 hive-server:
166   image: mrugankray/hive-server-sqoop:1.0
167   container_name: hive-server
168   depends_on:
169     - namenode
170     - datanode
171     - hive-metastore
172   env_file:
173     - ./hadoop.env
174   volumes:
175     - hadoop_hive_server_sqoop:/usr/lib/sqoop
176     - ./configs/hive_server_sqoop-env.sh:/usr/lib/sqoop/conf/sqoop-env.sh
177     - ./configs/hive_server/hive_server_bashrc.txt:/root/.bashrc
178     - ./configs/hive_server/sqoop-site.xml:/usr/lib/sqoop/conf/sqoop-site.xml
179     - ./configs/hive_server/ssh_config.conf:/etc/ssh/ssh_config
180   environment:
181     HIVE_CORE_CONF_javax_jdo_option_connectionURL: "jdbc:postgresql://hive-metastore/metastore"
182     SERVICE_PRECONDITION: "hive-metastore:9083"
  
```

```

181 HIVE_CORE_CONF_javax_jdo_option_connectionURL: "jdbc:postgresql://hive-metastore/metastore"
182 SERVICE_PRECONDITION: "hive-metastore:9083"
183 ports:
184   - 10000:10000
185 hive-metastore:
186   image: bde2020/hive:2.3.2-postgresql-metastore
187   container_name: hive-metastore
188   env_file:
189     - ./hadoop.env
190   command: /opt/hive/bin/hive --service metastore
191   environment:
192     SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 hive-metastore-postgresql:5432"
193   ports:
194     - "9083:9083"
195   depends_on:
196     - hive-metastore-postgresql
197   - hive-metastore-postgresql
198 hive-metastore-postgresql:
199   image: bde2020/hive-metastore-postgresql:2.3.0
200   container_name: hive-metastore-postgresql
201 control-center:
202   image: confluentinc/cp-enterprise-control-center:5.4.0
203   hostname: control-center
204   container_name: control-center
205   depends_on:
206     - zookeeper
207     - kafka-broker
208     - schema-registry
209   deploy:
210     restart_policy:
211       condition: on-failure
212       delay: 5s
213       max_attempts: 3
214       window: 120s
215   ports:
  
```

Réalisé par :

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

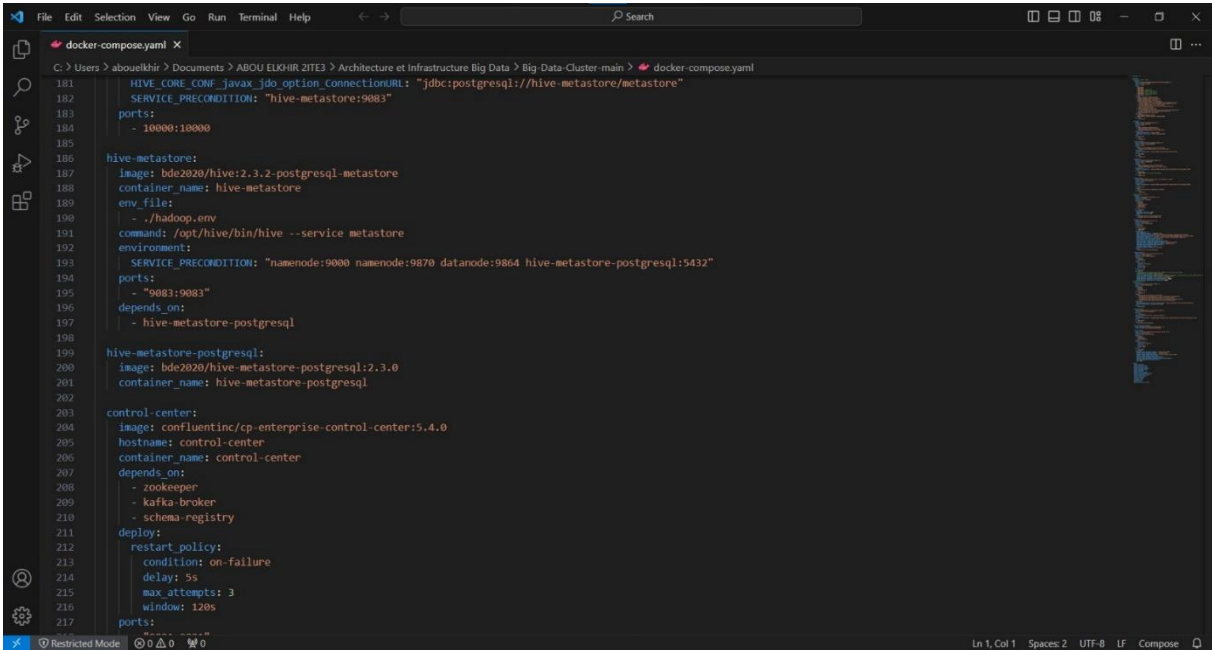


Figure 1.1 Code du fichier docker-compose pour la création des conteneurs docker

b. Exécution des conteneurs

Pour démarrer les conteneurs définis dans votre fichier **docker-compose.yml** et les exécuter en arrière-plan, utilisez la commande suivante : `docker-compose up -d`

Une fois les conteneurs lancés, vous pouvez lister les identifiants des conteneurs en cours d'exécution et leurs ports associés en utilisant la commande : `docker ps`

```

C:\Users\abouelkhir>docker ps

```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
f1b11b8f3e95	confluentinc/cp-enterprise-control-center:5.4.0	control-center	"/etc/confluent/dock..."	25 hours ago	Up 2 minutes	0.0.0.0:9021->9021/tcp
6d7e36565c82	confluentinc/cp-schema-registry:5.4.0	schema-registry	"/etc/confluent/dock..."	28 hours ago	Up 41 seconds	8081/tcp, 0.0.0.0:8083->8083/tcp
7e7f048174a5	confluentinc/cp-server:5.4.0	kafka-broker	"/etc/confluent/dock..."	28 hours ago	Up About a minute	0.0.0.0:9092->9092/tcp, 0.0.0.0:29092->29092/tcp
3852bb975387	mrugankray/hive-server-sqoop:1.0	hive-server	"entrypoint.sh /bin/..."	28 hours ago	Up About a minute	0.0.0.0:10000->10000/tcp, 10002/tcp
cd5662e42d2e	confluentinc/cp-zookeeper:5.4.0	zookeeper	"/etc/confluent/dock..."	28 hours ago	Up About a minute	2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp
f21981eb1b1a	bde2020/hive:2.3.2-postgresql-metastore	hive-metastore	"entrypoint.sh /opt/..."	28 hours ago	Up About a minute	10000/tcp, 0.0.0.0:9083->9083/tcp, 10002/tcp
13e26db850cf	mrugankray/nodemanager-python:1.0	nodemanager	"entrypoint.sh /run..."	28 hours ago	Up 2 minutes (healthy)	0.0.0.0:8042->8042/tcp, 0.0.0.0:19888->19888/tcp, 8088/tcp
90019b42a6ef	mrugankray/resourcemanager-python:1.0	resourcemanager	"entrypoint.sh /run..."	28 hours ago	Up About a minute (healthy)	8042/tcp, 0.0.0.0:8088->8088/tcp
3bd1e36cd869	mrugankray/namenode-spark-airflow-flume-zeppelin:1.1	namenode	"entrypoint.sh /sta..."	28 hours ago	Up 2 minutes (healthy)	0.0.0.0:3000->3000/tcp, 0.0.0.0:4040->4040/tcp, 0.0.0.0:8080->8080
-8082/tcp, 0.0.0.0:9000->9000/tcp, 0.0.0.0:9870->9870/tcp						
f83ce515deb	bde2020/hive-metastore-postgresql:2.3.0	hive-metastore-postgresql	"docker-entrypoint..."	28 hours ago	Up About a minute	5432/tcp
78bc9c746935	bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8	historyserver	"entrypoint.sh /run..."	28 hours ago	Up 2 minutes (healthy)	0.0.0.0:8188->8188/tcp
ef1e03c91d17	mrugankray/datanode-python:1.0	datanode	"entrypoint.sh /run..."	28 hours ago	Up 2 minutes (healthy)	0.0.0.0:9864->9864/tcp

Figure 1.2 Affichage de la liste des conteneurs

Pour accéder aux services dans un navigateur web, utilisez les URL suivantes :

OKACHA Najia

ASSOUMA Roukya

ABOUELKHIR Mohamed

- Zeppelin : <http://localhost:8082>
- Confluent : <http://localhost:9021>
- Spark : <http://localhost:8080>
- Airflow : <http://localhost:3000>
- Namenode : <http://localhost:9870>

Assurez-vous que les services sont correctement démarrés et que les ports spécifiés dans votre fichier **docker-compose.yml** ne sont pas utilisés par d'autres applications sur votre système.

c. Exécution d'un conteneur spécifique

Pour accéder au shell d'un conteneur spécifique en mode bash, utilisez la commande suivante, en remplaçant **id_container** par l'identifiant du conteneur souhaité :

`docker exec -it id_container /bin/bash`

```
C:\Users\abouelkhir>docker exec -it 3852bb975387 /bin/bash
root@3852bb975387:/# hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/opt/hive/conf/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive>
```

Figure 1.3 Exécution d'un conteneur spécifique

3. Installation et Configuration de Tableau

Dans le cadre de ce projet, nous utilisons Tableau Desktop. Suivez ces étapes pour installer et configurer Tableau Desktop sur votre machine.

a. Téléchargement de Tableau Desktop :

- Rendez-vous sur le site officiel de Tableau : [Tableau Download](#)

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

+ a b l e a u

from Salesforce

Why Tableau

Products

Solutions

Resources

Partners

PRICING

SIGN IN

TRY NOW

BUY NOW

Tableau Desktop: Start your free 14-day trial

Almost there!

It only takes 15 seconds to fill out. If you're already registered,

[sign in.](#)

First Name

Last Name

Figure 1.4 Site de téléchargement de tableau desktop

- Téléchargez la version appropriée de Tableau Desktop pour votre système d'exploitation (Windows ou Mac).
- Installez Tableau en suivant les instructions fournies lors du processus d'installation.

b. Activation de Tableau :

- Lors du premier lancement, Tableau Desktop vous demandera d'activer le produit. Entrez votre clé de licence si vous en avez une, ou choisissez la version d'essai.

c. Connexion à Hive depuis Tableau :

Pour connecter Tableau à Hive, vous pouvez utiliser le pilote Hive Tableau Connector. Suivez ces étapes générales pour effectuer cette connexion.

- Téléchargez et installez un pilote compatible avec Hive sur votre machine. Un exemple courant est le pilote Hive Tableau Connector, que vous pouvez trouver sur le site de CDATA : [Hive Tableau Connector](#).

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

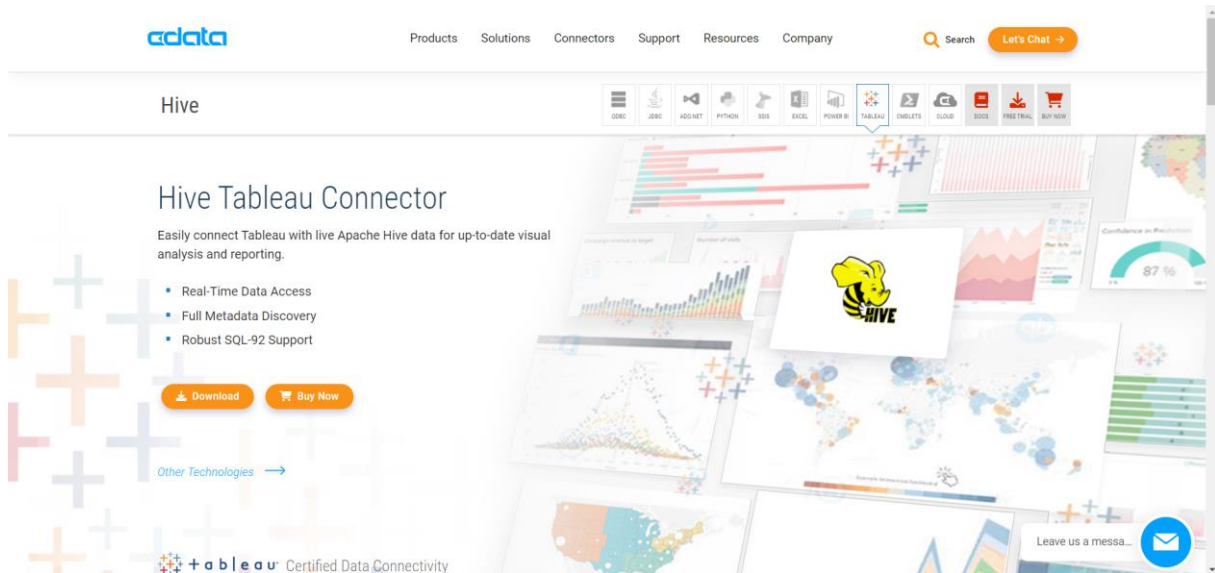
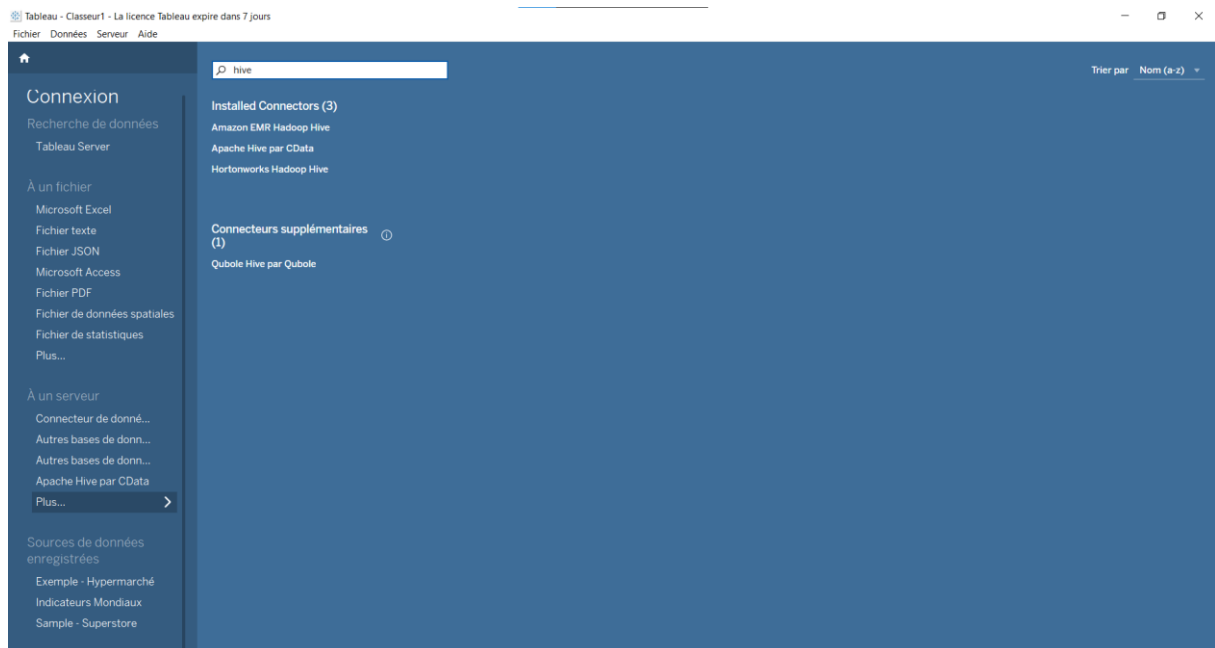


Figure 1.5 Site de téléchargement de Hive Tableau connector

- Après l'installation du pilote, configurez-le en fournissant les détails de connexion à votre cluster Hive, tels que l'adresse du serveur Hive, le port, le nom d'utilisateur, le mot de passe, etc.



- Dans Tableau Desktop, sous l'onglet "Connexion a un serveur", choisissez "Apache Hive par CData" comme type de connexion.

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

- Sélectionnez le pilote pour Hive que vous avez installé.
- Entrez les informations de connexion nécessaires, telles que le nom du serveur Hive, le port, le nom d'utilisateur et le mot de passe.

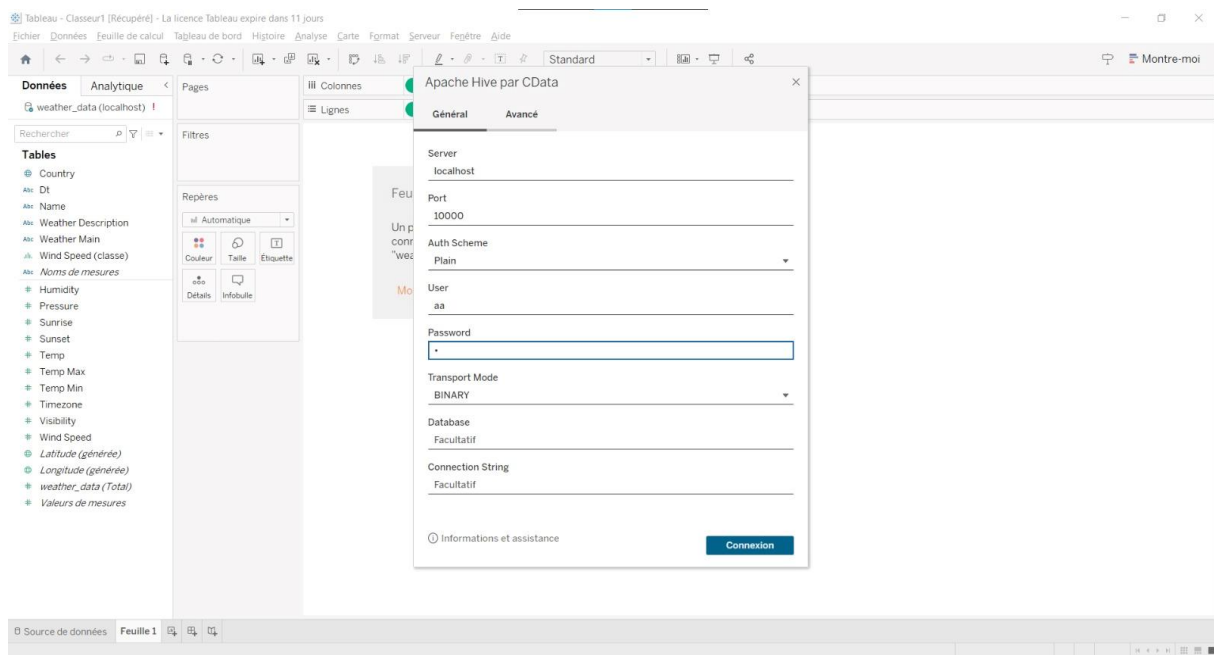


Figure 1.6 Etablissement de la connexion entre Hive et Tableau

- En fonction du pilote que vous utilisez, il peut y avoir des options supplémentaires à configurer. Consultez la documentation du pilote pour plus de détails.
- Après avoir configuré les paramètres de connexion, cliquez sur le bouton "Connecter" pour établir la connexion entre Tableau et Hive.

Note : Par défaut, aucun utilisateur n'est configuré sur Hive. Vous devez définir ici un utilisateur et un mot de passe que vous utiliserez à chaque fois que vous établirez la connexion.

d. Sélectionner la Table ou la Vue Hive :

Une fois connecté, vous pourrez voir les bases de données Hive disponibles. Sélectionnez la base de données, puis choisissez la table ou la vue Hive que vous souhaitez analyser dans Tableau. Pour le moment vous il n'y a aucune table dans la base de données par défaut de hive.

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

e. Configuration des Connexions de Données :

Vous pouvez configurer les connexions de données pour qu'elles soient dynamiques et mises à jour automatiquement en fonction des changements dans vos sources de données.

III. Création du Dashboard pour la visualisation

1. Obtention de la clé d'API OpenWeatherMap

- Accédez au site web d'OpenWeatherMap: [OpenWeatherMap](https://openweathermap.org/).

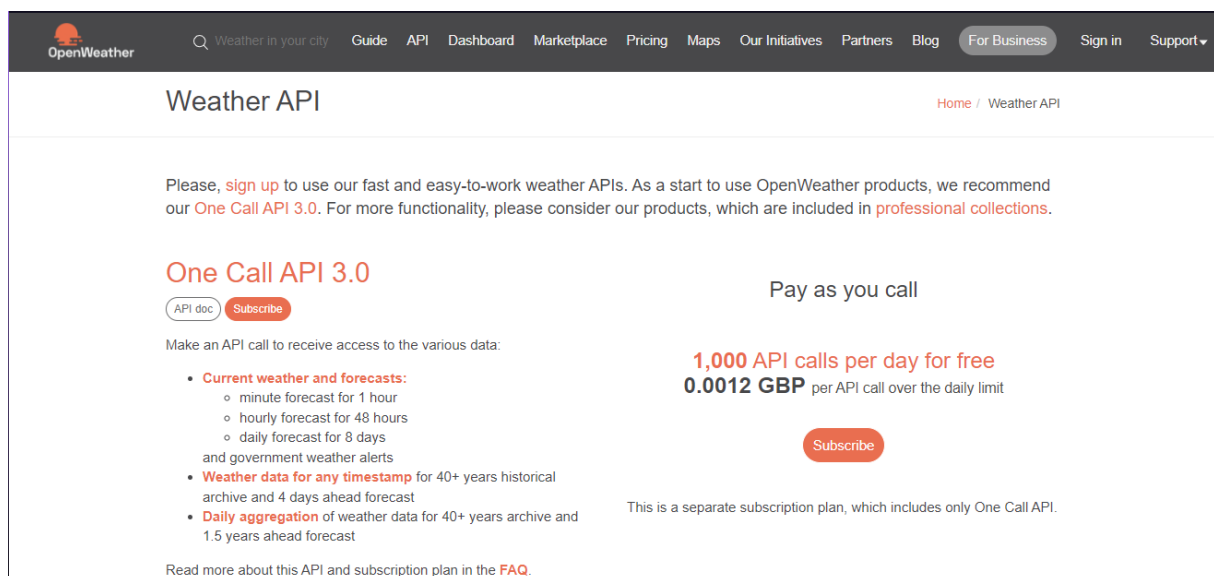


Figure 3.1 Site de OpenWeather

- Cliquez sur "Sign Up" (S'inscrire) pour créer un nouveau compte.
- Connectez-vous à votre compte OpenWeatherMap en utilisant les identifiants que vous avez créés.
- Une fois connecté, recherchez la section "My API keys" (clés API) dans votre espace utilisateur.

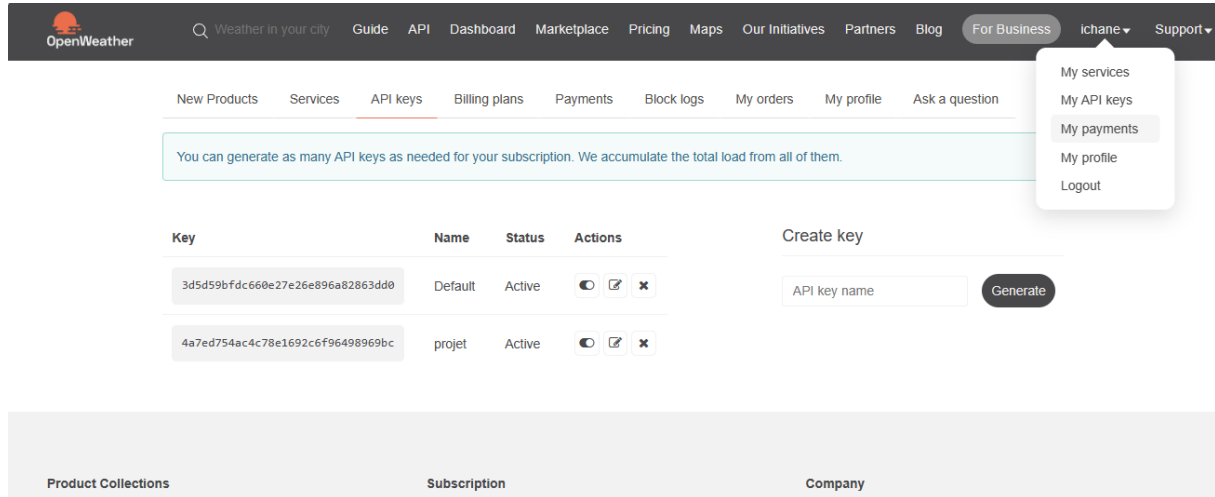


Figure 3.2 Obtention de la clé d'api

- Une fois la clé générée, copiez-la. Cette clé sera utilisée dans vos requêtes API pour identifier votre compte et autoriser l'accès aux données météorologiques.

2. Collecte des Données Météorologiques depuis l'api vers un topic Kafka

Pour collecter des données météorologiques depuis l'API OpenWeatherMap et les publier sur un topic Kafka en utilisant Python, vous pouvez utiliser la bibliothèque requests pour effectuer la requête API et la bibliothèque kafka-python pour publier les données sur Kafka. Assurez-vous d'installer ces bibliothèques en utilisant la commande suivante si elles ne sont pas déjà installées.

- Créer un notebook python sur zeppelin et coller le code de collecte des données depuis l'api. Vous trouverez le code dans le fichier kafka_stream dans le dossier du projet.
- Assurer vous de remplacer la clé d'api par la vôtre.
- Installer le package kafka-python avec le code suivant :

```
import subprocess

# Replace 'kafka-python' with the package you want to install
package_to_install = 'kafka-python'

# Use subprocess to run the pip install command
subprocess.call(['pip', 'install', package_to_install])
```

- Assurez-vous que Kafka est en cours d'exécution et accessible depuis Zeppelin.

OKACHA Najia

ASSOUMA Roukya

ABOUELKHIR Mohamed

Note : en fonction de votre api : gratuit ou payant vous pouvez avoir accès à différents types de données. Assurer vous donc d'adapter le code en fonction de la structure des données météorologiques que vous souhaitez collecter.

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed



Figure 3.3 Code de collecte des données depuis l'api vers un topic Kafka

OKACHA Najia

ASSOUMA Roukya

ABOUELKHIR Mohamed

Les données /les messages peuvent être visualiser au niveau du control center à l'adresse ([http://localhost: 9021](http://localhost:9021))

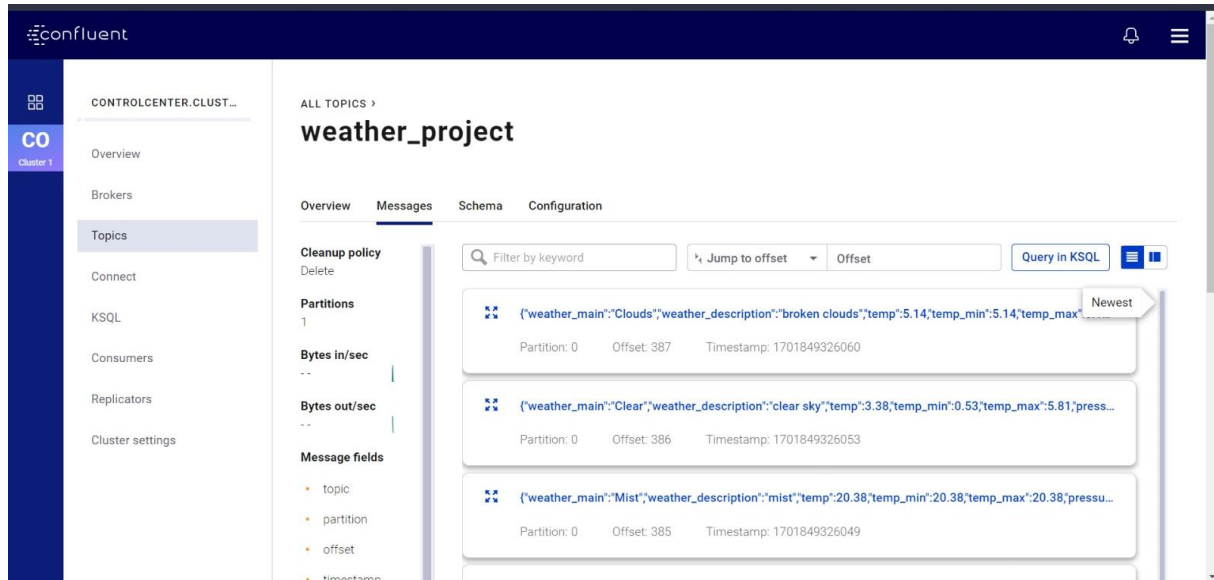


Figure 3.4 Visualisation des données du topic Kafka sur le control center

3. Traitement et Stockage des Données avec Spark et Hive

Pour effectuer le traitement et le stockage des données avec Spark et Hive en utilisant PySpark dans Zeppelin, vous pouvez suivre ces étapes.

Assurez-vous que votre environnement est correctement configuré pour utiliser PySpark avec Zeppelin et que tout fonctionne correctement.

Le code à la Figure 3.5 orchestre la réception de données depuis le topic Kafka, lesquelles sont soumises à des opérations de traitement en temps réel à l'aide de Spark Streaming. Une fois ces traitements effectués, les résultats obtenus sont consignés et enregistrés dans Hive, permettant ainsi une structuration et une conservation optimales de ces données traitées au sein de la base de données Hive.

- Créer un nouveau notebook PySpark sur zeppelin et coller le code à exécuter. Vous trouverez le code dans le fichier spark_stream dans le dossier du projet.
- Avant de lancer l'exécution de votre code (spark_stream) créer une table weather_data dans Hive, qui va contenir les données. Par défaut cette table sera enregistré dans la base de données default: Ci-dessous le script de création de la table.

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

```
CREATE TABLE IF NOT EXISTS weather_data (
    weather_main STRING,
    weather_description STRING,
    temp DOUBLE,
    temp_min DOUBLE,
    temp_max DOUBLE,
    pressure INT,
    humidity INT,
    visibility INT,
    wind_speed DOUBLE,
    dt STRING,
    country STRING,
    sunrise STRING,
    sunset STRING,
    timezone INT,
    name STRING
)
STORED AS ORC ;
```

Par la suite lancer l'exécution de votre code.

The screenshot shows the Zeppelin Notebook interface. At the top, there's a blue header bar with the Zeppelin logo, the word "Notebook", and a "Job" tab. A search bar is on the right. Below the header, the main area is titled "spark streaming" and contains a code editor with a PySpark script. The script configures a SparkSession, sets Kafka details, defines a schema, and reads data from Kafka. On the right side of the code editor, there's a status bar showing "RUNNING 0%" and some icons. The bottom of the interface shows a snippet of the script: "# Read data from Kafka" and "df = spark.readStream \".

Figure 3.5 Code pour récupérer les données du topic et les envoyer vers Hive

Pour vérifier que les données ont été correctement stockées dans Hive, accéder à l'interface de Hadoop.

Réalisé par :

OKACHA Najia

ASSOUMA Roukèya

ABOUELKHIR Mohamed

Hadoop	Overview	Datanodes	Datanode Volume Failures	Snapshot	Startup Progress	Utilities
Browse Directory						
/user/hive/warehouse/weather_data2/						
Show 25 entries						
Permission	Owner	Group	Size	Last Modified	Replication	Block Size
drwxr-xr-x	root	supergroup	0 B	Dec 05 19:06	0	0 B
-rw-r--r--	root	supergroup	2.41 KB	Dec 05 19:05	3	256 MB
-rw-r--r--	root	supergroup	337 B	Dec 05 18:59	3	256 MB
-rw-r--r--	root	supergroup	2.72 KB	Dec 05 19:04	3	256 MB
-rw-r--r--	root	supergroup	2.28 KB	Dec 05 19:06	3	256 MB
-rw-r--r--	root	supergroup	2.8 KB	Dec 05 18:55	3	256 MB
-rw-r--r--	root	supergroup	2.72 KB	Dec 05 19:06	3	256 MB
-rw-r--r--	root	supergroup	2.65 KB	Dec 05 19:05	3	256 MB
-rw-r--r--	root	supergroup	2.28 KB	Dec 05 19:04	3	256 MB

Figure 3.6 Vérification au niveau du namenode de l'ajout des données sur Hive

Par la suite vous pouvez vérifier dans la base de données pour s'assurer que les données ont bien été enregistré :

```
hive> select * from weather_data2;
OK
Clouds overcast clouds 17.94 17.94 19.1 1018 59 10000 1.79 2023-12-05T12:05:47+01:00 MA 2023-12-05 07:19:54 2023-12-05 17:22:12 3600 Casablanca
Clear clear sky 31.61 31.61 31.61 1012 59 10000 2.82 2023-12-05T12:09:18+01:00 TG 2023-12-05 05:52:15 2023-12-05 17:39:14 0 Lomé
Clouds broken clouds 5.98 4.97 6.94 1007 91 10000 3.6 2023-12-05T12:04:58+01:00 GB 2023-12-05 07:48:48 2023-12-05 15:53:24 0 London
Clouds overcast clouds 20.6 20.6 20.6 1015 51 10000 6.88 2023-12-05T12:09:18+01:00 MA 2023-12-05 07:22:38 2023-12-05 17:26:38 3600 El Jadida
Clouds broken clouds 11.4 9.74 12.69 1009 73 10000 3.09 2023-12-05T12:04:45+01:00 IT 2023-12-05 06:21:56 2023-12-05 15:39:19 3600 Trevi
Rain light rain 9.6 8.12 10.86 1014 58 10000 3.6 2023-12-05T12:08:48+01:00 JP 2023-12-04 21:35:32 2023-12-05 07:27:52 32400 Tokyo
Rain moderate rain 14.51 13.22 15.64 1011 94 4828 5.14 2023-12-05T12:09:20+01:00 US 2023-12-05 15:41:22 2023-12-06 00:18:51 -28800 Seattle
Clouds scattered clouds 21.38 21.38 21.38 1013 68 6000 2.06 2023-12-05T12:04:22+01:00 EG 2023-12-05 04:36:18 2023-12-05 14:55:01 7200 Al 'Atabah
Clear clear sky 6.15 4.59 6.88 1019 72 10000 2.06 2023-12-05T12:04:03+01:00 ES 2023-12-05 07:22:19 2023-12-05 16:48:34 3600 Madrid City Center
Clouds overcast clouds 12.4 12.4 12.4 1019 25 10000 1.5 2023-12-05T12:05:22+01:00 MA 2023-12-05 07:18:28 2023-12-05 17:21:36 3600 Ifrane
Clouds broken clouds 17.43 17.18 19.99 1015 59 6000 2.06 2023-12-05T11:59:03+01:00 MA 2023-12-05 07:19:54 2023-12-05 17:22:12 3600 Casablanca
few clouds 32.07 32.07 32.07 1013 66 10000 4.12 2023-12-05T11:59:03+01:00 TG 2023-12-05 05:52:15 2023-12-05 17:39:14 0 Lomé
Rain light rain 5.91 4.97 6.94 1007 91 9000 3.09 2023-12-05T11:59:03+01:00 GB 2023-12-05 07:48:48 2023-12-05 15:53:24 0 London
Clouds overcast clouds 20.6 20.6 20.6 1015 51 10000 6.88 2023-12-05T11:59:04+01:00 MA 2023-12-05 07:22:38 2023-12-05 17:26:38 3600 El Jadida
Clouds broken clouds 11.34 9.74 12.69 1009 73 7000 3.09 2023-12-05T11:58:29+01:00 IT 2023-12-05 06:21:56 2023-12-05 15:39:19 3600 Trevi
Rain moderate rain 9.6 8.12 10.86 1014 58 10000 3.6 2023-12-05T11:58:02+01:00 JP 2023-12-04 21:35:32 2023-12-05 07:27:52 32400 Tokyo
Rain moderate rain 14.52 13.22 15.64 1011 94 8047 6.17 2023-12-05T11:59:05+01:00 US 2023-12-05 15:41:22 2023-12-06 00:18:51 -28800 Seattle
Clouds scattered clouds 20.38 20.38 20.38 1014 77 6000 1.54 2023-12-05T11:59:05+01:00 EG 2023-12-05 04:36:18 2023-12-05 14:55:01 7200 Giza
Clear clear sky 5.92 4.59 6.88 1019 72 10000 2.06 2023-12-05T11:54:46+01:00 ES 2023-12-05 07:22:19 2023-12-05 16:48:34 3600 Madrid
Clouds overcast clouds 12.4 12.4 12.4 1019 25 10000 1.5 2023-12-05T11:59:06+01:00 MA 2023-12-05 07:18:28 2023-12-05 17:21:36 3600 Ifrane
Clouds overcast clouds 17.94 17.94 19.1 1018 59 10000 1.79 2023-12-05T12:05:47+01:00 MA 2023-12-05 07:19:54 2023-12-05 17:22:12 3600 Casablanca
Clear clear sky 31.61 31.61 31.61 1012 59 10000 2.82 2023-12-05T12:09:18+01:00 TG 2023-12-05 05:52:15 2023-12-05 17:39:14 0 Lomé
Clouds broken clouds 5.98 4.97 6.94 1007 91 10000 3.6 2023-12-05T12:04:58+01:00 GB 2023-12-05 07:48:48 2023-12-05 15:53:24 0 London
Clouds overcast clouds 20.6 20.6 20.6 1015 51 10000 6.88 2023-12-05T12:09:18+01:00 MA 2023-12-05 07:22:38 2023-12-05 17:26:38 3600 El Jadida
Clouds broken clouds 11.4 9.74 12.69 1009 73 10000 3.09 2023-12-05T12:04:45+01:00 IT 2023-12-05 06:21:56 2023-12-05 15:39:19 3600 Trevi
Rain light rain 9.6 8.12 10.86 1014 58 10000 3.6 2023-12-05T12:08:48+01:00 JP 2023-12-04 21:35:32 2023-12-05 07:27:52 32400 Tokyo
Rain moderate rain 14.51 13.22 15.64 1011 94 4828 5.14 2023-12-05T12:09:20+01:00 US 2023-12-05 15:41:22 2023-12-06 00:18:51 -28800 Seattle
Clouds scattered clouds 21.38 21.38 21.38 1013 68 6000 2.06 2023-12-05T12:04:22+01:00 EG 2023-12-05 04:36:18 2023-12-05 14:55:01 7200 Al 'Atabah
Clear clear sky 6.15 4.59 6.88 1019 72 10000 2.06 2023-12-05T12:04:03+01:00 ES 2023-12-05 07:22:19 2023-12-05 16:48:34 3600 Madrid City Center
Clouds overcast clouds 12.4 12.4 12.4 1019 25 10000 1.5 2023-12-05T12:05:22+01:00 MA 2023-12-05 07:18:28 2023-12-05 17:21:36 3600 Ifrane
Time taken: 0.261 seconds, Fetched: 30 row(s)
```

Figure 3.7 Vérification au niveau de Hive

ABOUELKHIR Mohamed

Tableau - Classeur1 - La licence Tableau expire dans 12 jours

Fichier Données Serveur Fenêtre Aide

Connexions Ajouter

- localhost
Apache Hive par CDATA

Table

- weather_data

Nouvelle requête SQL personnalisée


Nouvelle union

Nouvelle extension de table

weather_data (localhost)

Connexion En direct Extraire Filtres 0 Ajouter

weather_data



Vous avez besoin de données supplémentaires ?
Faites glisser les tables ici pour les relier. En savoir plus

weather_data 15 champs 95 lignes											
>	weather_data	weather_data	weather_data	weather_data	weather_data	weather_data	weather_data	weather_data	weather_data	weather_data	weather_data
	p Min	Temp Max	Pressure	Humidity	Visibility	Wind Speed	Dt	Country	Sunrise	Sunset	Timezone
Détails de la table	281.8200	284.0100	1015.0000	56.0000	10 000.00	3.09000	2023-12-05T09:10:44+01:00	JP	1701725732	1701761272	32400
	282.0800	284.3700	1015.0000	54.0000	10 000.00	4.12000	2023-12-05T09:14:21+01:00	JP	1701725691	1701761242	32400
	270.9200	272.7700	997.0000	84.0000	10 000.00	6.17000	2023-12-05T09:03:43+01:00	DE	1701759574	1701788065	3600
	274.6800	278.9200	1018.0000	80.0000	10 000.00	1.79000	2023-12-05T09:10:43+01:00	ES	1701760939	1701794914	3600
	282.0600	282.0600	1019.0000	35.0000	10 000.00	1.19000	2023-12-05T09:14:22+01:00	MA	1701760228	1701796896	3600
	287.2500	290.1400	1015.0000	67.0000	6 000.00	1.03000	2023-12-05T09:14:18+01:00	MA	1701760794	1701796932	3600

Source des données Feuille 1

18

ABOUELKHIR Mohamed

- | Name / date | |
|-------------|--|
| Al 'Atabah | |
| Casablan.. | |
| El Jadida | |
| Ifrane | |
| Lomé | |
| London | |
| Madrid | |
| Ma.. | |
| Seattle | |
| Sol | |
| Tokyo | |
| Trevi | |

19

ABOUELKHIR Mohamed



Allez dans l'onglet "Tableau de Bord" pour créer une vue d'ensemble qui combine plusieurs visualisations.

ABOUELKHIR Mohamed



OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

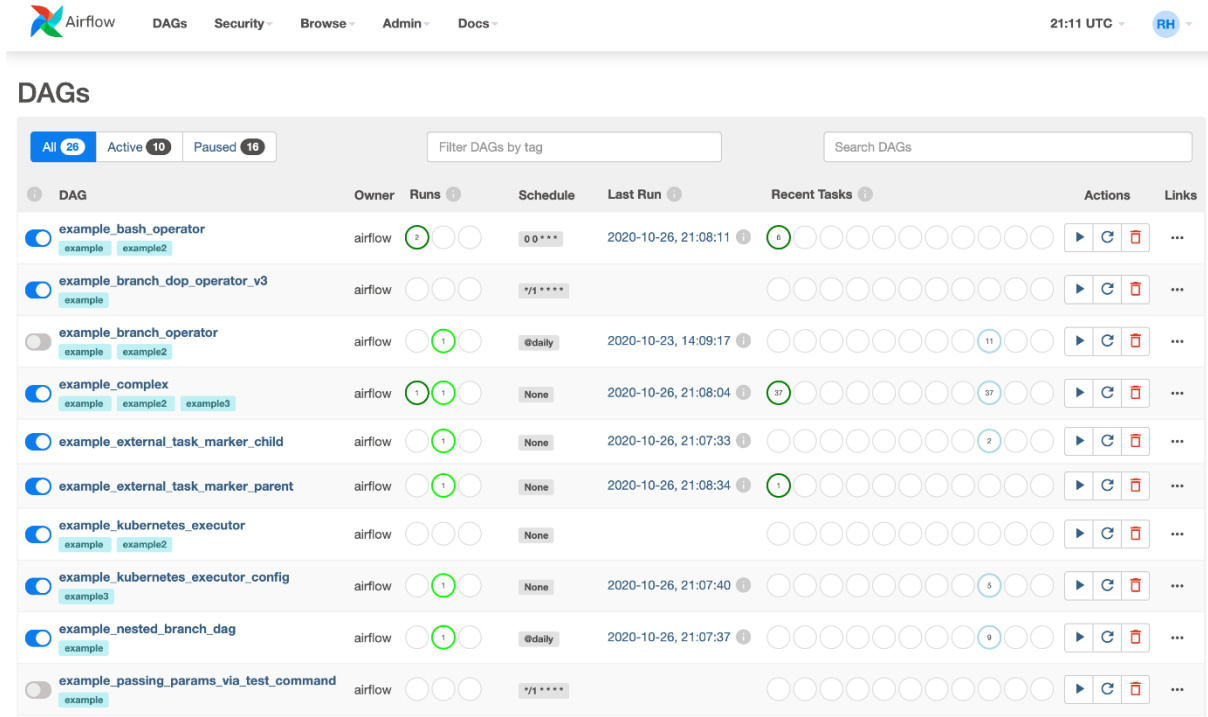


Figure 3.12 Interface utilisateur de Airflow

b. Création et exécution du dag

- Utilisez votre éditeur de code préféré (comme VSCode, PyCharm, etc.).
- Créez un nouveau dossier appelé "dags" à l'emplacement où vous souhaitez stocker vos DAGs.
- À l'intérieur du dossier "dags", créez un fichier Python pour votre DAG, par exemple, "dag.py".
- Dans ce fichier, écrivez le code décrivant les différentes étapes du processus.

Vous trouverez le code dans le fichier dag.py dans le dossier du projet, copier et coller le code dans votre fichier et faites les installations nécessaires.

- Faites toutes les installations de packages nécessaires avec pip ou si vous utiliser PyCharm à partir de l'interpréteur.
- Dans le code du dag modifier les adresses des notebooks selon les id

Réalisé par :

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

```

1  > import ...
2
3  # Default DAG arguments
4  default_args = {
5      'owner': 'airflow',
6      'depends_on_past': False,
7      'start_date': days_ago(1),
8      'retries': 1,
9      'retry_delay': timedelta(minutes=5),
10 }
11
12 # usage
13 def execute_zeppelin_notebook_1(**kwargs):
14     # Zeppelin API endpoint to execute a notebook
15     zeppelin_api_url = 'http://localhost:8082/#/notebook/2JFYH4S8U'
16
17     # Make a POST request to execute the Zeppelin notebook
18     response = requests.post(zeppelin_api_url)
19
20     # Check if the request was successful (HTTP status code 200)
21     if response.status_code == 200:
22         print("Zeppelin notebook executed successfully 1.")
23     else:
24         print(f"Failed to execute Zeppelin notebook 1. Status code: {response.status_code}")
25         print(response.text)
26
27 execute_zeppelin_notebook_1()
    
```

```

31 # usage
32 def execute_zeppelin_notebook_2(**kwargs):
33     # Zeppelin API endpoint to execute a notebook
34     zeppelin_api_url = 'http://localhost:8082/#/notebook/2JGNWB8KA'
35
36     # Make a POST request to execute the Zeppelin notebook
37     response = requests.post(zeppelin_api_url)
38
39     # Check if the request was successful (HTTP status code 200)
40     if response.status_code == 200:
41         print("Zeppelin notebook executed successfully 2.")
42     else:
43         print(f"Failed to execute Zeppelin notebook 2. Status code: {response.status_code}")
44         print(response.text)
45
46 # Create the main DAG
47 default_args = {
48     'owner': 'airflow',
49     'depends_on_past': False,
50     'start_date': datetime(year=2023, month=12, day=1),
51     'retries': 1,
52     'retry_delay': timedelta(minutes=5),
53 }
54 # Define the DAG
55 execute_zeppelin_notebook_1()
    
```

Réalisé par :

OKACHA Najia

ASSOUMA Roukya

ABOUELKHIR Mohamed

```

# Create the main DAG
48 default_args = {
49     'owner': 'airflow',
50     'depends_on_past': False,
51     'start_date': datetime(year=2023, month=12, day=1),
52     'retries': 1,
53     'retry_delay': timedelta(minutes=5),
54 }
55
56 # Define the DAG
57 dag = DAG(
58     dag_id='weather_project_dag',
59     default_args=default_args,
60     description='DAG to fetch weather data and send to Kafka',
61     schedule_interval='@daily',
62 )
63
64 with dag:
65     # Use the PythonOperator to execute the Zeppelin notebook
66     kafka_stream = PythonOperator(
67         task_id='kafka_stream',
68         python_callable=execute_zeppelin_notebook_1,
69         provide_context=True, # Pass the context to the Python function
70     )
71
72     spark_stream = PythonOperator(
73         task_id='spark_stream',
74         python_callable=execute_zeppelin_notebook_1,
75     )
76
77     kafka_stream >> spark_stream
78
79 execute_zeppelin_notebook_1()
    
```

Figure 3.13 Code du dag du projet

- Assurez-vous qu'Airflow est en cours d'exécution et lancer l'exécution de votre dag.
- Airflow va automatiquement détecter et charger votre DAG.

Réalisé par :

OKACHA Najia

ASSOUMA Roukéya

ABOUELKHIR Mohamed

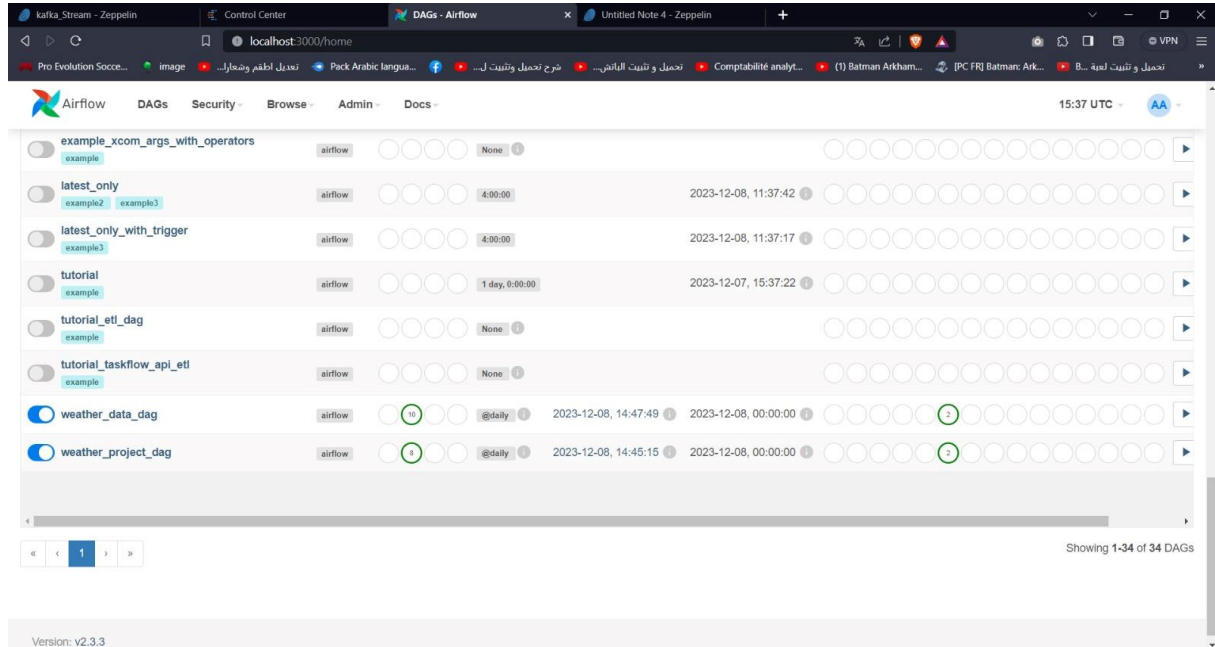


Figure 3.14 Vérification de l'ajout du nouveau dag à la liste des dags

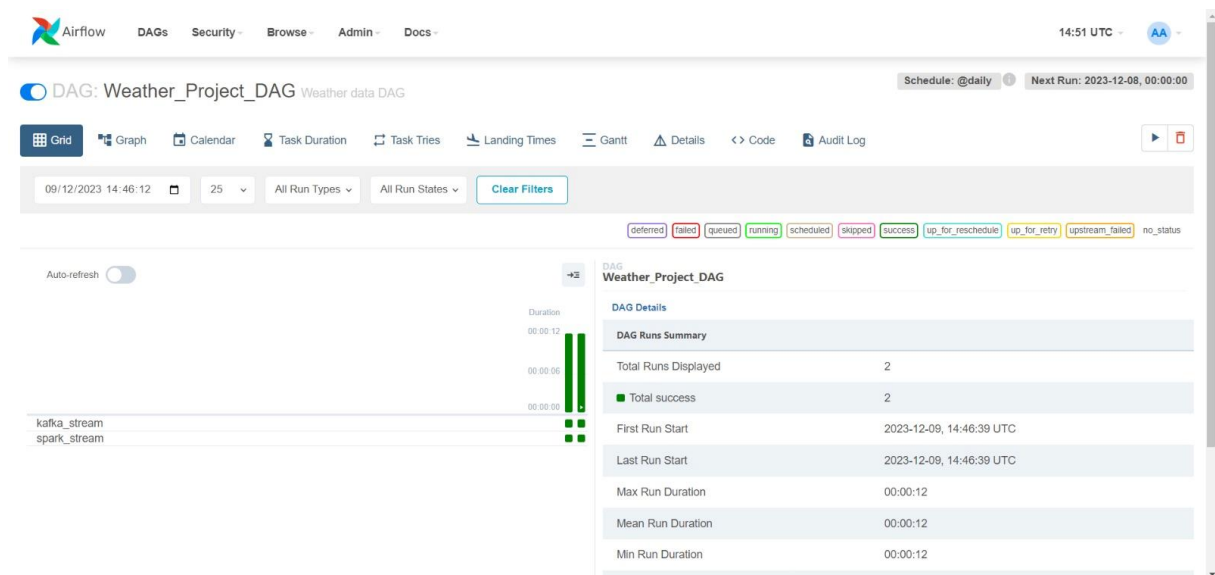


Figure 3.15 Visualisation de l'exécution du dag créer

OKACHA Najia

ASSOUMA Roukya

ABOUELKHIR Mohamed

Si vous avez planifié le processus de collecte, de traitement et de stockage des données à intervalles réguliers avec Apache Airflow, vous pouvez configurer Tableau pour actualiser automatiquement les données à partir de Hive.